Some Applications of Distributed Signal Processing

Sergio Valcarcel Macua, Santiago Zazo

***Abstract*— In this work we review some earlier distributed algorithms developed by the authors and collaborators, which are based on two different approaches, namely, distributed moment estimation and distributed stochastic approximations. We show applications of these algorithms on image compression, linear classification and stochastic optimal control. In all cases, the benefit of cooperation is clear: even when the nodes have access to small portions of the data, by exchanging their estimates, they achieve the same performance as that of a centralized architecture, which would gather all the data from all the nodes.**

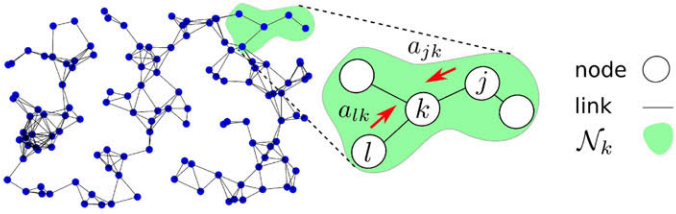

Fig. 1. Example of networks considered in this work.

I. Introduction

In 'distributed' algorithms, each node in a network is able to process information locally and it interacts with its neighbors in order to improve its performance (e.g., [1], [2]). This is different from a a centralized architecture, in which the nodes transmit and route their data to a central station that performs the learning process. Large networks can benefit from distributed schemes for several reasons: they are more robust to node failures than a centralized architecture, the per-node communication cost depends on the size of the neighborhood rather than on the size of the network, and privacy is respected because no data samples are exchanged.

II. Preliminaries

Consider a network of N cooperative agents with arbitrary topology. The network is modeled by a graph where the nodes are the agents, and edges represent the communication links (see Figure 1). We assume the graph is connected (i.e., there is at least one path between any pair of nodes). The agents want to estimate some parameter vector, w^o (e.g., the sample estimate of a sufficient statistic of the data, or the minimizer of some objective function). If each individual agent has only access to a subset of the data, its individual learning process may be biased. Nevertheless, by cooperating with its neighbors each node can approach the same performance as the one of a centralized architecture.

Many distributed algorithms can be summarized by the following two steps [3]:

$$\phi_{k,i} = w_{k,i-1} - \mu \cdot f_k(w_{k,i-1}) \tag{1a}$$

$$w_{k,i} = \sum_{l \in \mathcal{N}_k} a_{lk} \phi_{l,i} \tag{1b}$$

where $w_{k,i}$ denotes the estimate of the parameter of interest at time i by node k, $\phi_{k,i}$ is some local function that depends on the problem at hand (e.g., the gradient of a local objective function in a distributed optimization problem), $\mathcal{N}_k$ denotes the neighborhood of node k (i.e., all the nodes that can share information with node k, including k itself). and a_{lk} is the non-negative weight given by node k to the information shared by node l. In (1a), each node processes information locally through a stochastic approximation with step-size μ. Then, in (1b), each node combines the estimates of its neighbors (including itself) with the corresponding weights. Since the network is connected, information is diffused across the network and each intermediate estimate evolves influenced by the learning process of every other node.

III. Distributed Estimation

Let us say that each agent k wants to estimate the sample mean of the observations of all the agents across the network, but it only has access to its own observation y_k. This problem is solved by the average consensus algorithm (see, e.g., [2]), which is a particular case of (1b)–(1a) without update step:

$$\phi_{k,i-1} = \sum_{l \in \mathcal{N}_k} a_{lk} w_{l,i-1}, \quad w_{k,i} = \phi_{k,i-1} \tag{2}$$

where we set the initial estimate equal to the local observation (i.e., $w_{k,0} = y_k$).

Assume that observations are scalar valued, i.e., $y_k \in \Re$. Introduce the network vector $w_i = [w_{1,i}, \ldots, w_{N,i}]^\top$, with entries the individual estimates of all the nodes, and collect the weights a_{lk} into a combination matrix A of size $N \times N$. Then, we can express (2) as a network recursion: $w_i = A^\top w_{i-1}$. When A satisfies the following three conditions:

$$\rho\left(A^\top - \frac{1}{N}\mathbb{1}_N \mathbb{1}_N^T\right) < 1, \quad A^\top \mathbb{1}_N = \mathbb{1}_N, \quad A\mathbb{1}_N = \mathbb{1}_N \tag{3}$$

(where $\mathbb{1}_N$ denotes the vector of ones with length N, and $\rho(\cdot)$ denotes the spectral radius of a matrix), then, every node will

approach the global sample mean:

$$\lim_{i\to\infty} w_i = \lim_{i\to\infty} A^i w_0 \approx \frac{1}{N}\sum_{k=1}^{N} x_k \triangleq \mathbb{1}_N \cdot w^o \qquad (4)$$

A. Distributed averaging of time series of different length

Let us assume the existence of a *global* data matrix Y with T data points, i.e., the column vectors $y_{k,i} \in \Re^D$, which are not available at any one location in the system, but rather they are sparsely available at each node. This is depicted in Figure 2, where each node k has the knowledge of a block of t_k data points. For this system model, the global average

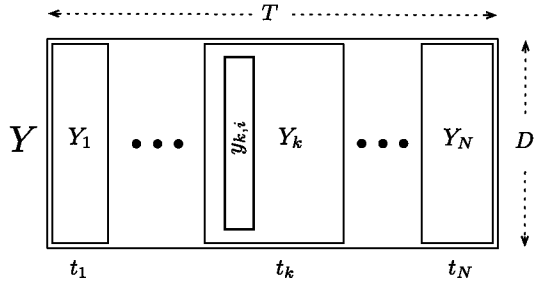

Fig. 2. Global data matrix Y with a total of T samples, distributed unevenly over N nodes

is given by $m^o = \frac{1}{T}\sum_{k=1}^{N}\sum_{i=1}^{t_k} y_{k,i}$. We are interested in computing m^o in a distributed fashion.

In references [4], [5], Belanovic and ourselves propose to apply the average consensus algorithm (2) to approximate the global mean when each node has a different number of samples. The method consists of three steps. introduce the *c*onsensus operator $\chi(w) \triangleq \lim_{i\to\infty} A^i w$, such that $\chi(w) = [\chi_1(w), \ldots, \chi_N(w)]^\top$ is the vector of estimates across the network, where $\chi_k(w)$ is the estimate of w^o at node k (we write $\chi_k(w) \approx w^o$, $1 \le k \le N$ to emphasize that these terms are asymptotically equal, i.e., when $i \to \infty$). Let $t = [t_1, \ldots, t_N]^\top$ be the vector with the number of samples available at every node. In the first step of the algorithm, by applying (2), every node asymptotically achieves consensus in the mean number of samples per node, that we denote as t^o:

$$\chi_k(t) \triangleq \widehat{t}_k^o \approx \frac{1}{N}\sum_{k=1}^{N} t_k = \frac{1}{N} T \triangleq t^o \qquad (5)$$

Next, each node locally adds all its local samples up, and weights the result with $1/\widehat{t}_k^o$, getting $m_k = \frac{1}{\widehat{t}_k^o}\sum_{i=1}^{t_k} y_{k,i}$. Finally, all the nodes run a second consensus loop on its local m_k. Since $T = Nt^o$, after running consensus, every node obtains an estimate of the global average:

$$\chi_k(m) \triangleq \widehat{m}_k^o \approx \frac{1}{N}\sum_{k=1}^{N} m_k \approx \frac{1}{N}\sum_{k=1}^{N}\frac{1}{t^o}\sum_{i=1}^{t_k} y_{k,i} \triangleq m^o \qquad (6)$$

where $m = [m_1, \ldots, m_N]^\top$.

B. Application 1: Distributed image compression with PCA

Principal Component Analysis (PCA) is a classic feature extraction method, which finds a subspace spanned by the directions of maximum variance of the data. If we take the P dimensions of maximum variance, remove the $D - P$ directions of minimum variance, and project the data onto this subspace with smaller dimensionality (usually named principal subspace), then much of the variance of the data will be preserved. The centralized PCA requires all the data samples $y_{k,i}$, from all the nodes, to be gathered at a fusion center. With this access to the full data set (i.e., the whole matrix Y), the fusion center can obtain the principal subspace $B \in \Re^{D\times P}$ (where $P \ll D$) directly by taking the P dominant eigenvectors of the sample covariance matrix, which is

$$\Sigma^o = \frac{1}{T}\sum_{k=1}^{N}\sum_{i=1}^{t_k}(y_{i,k} - m^o)(y_{i,k} - m^o)^\top \qquad (7)$$

One approach to implement PCA in a distributed fashion is to estimate Σ^o by using the average consensus technique–described above–over the local covariance matrices. The method that the authors and Belanovic proposed in [5], [6] follows. First, the network runs two consensus loop to estimate the average number of samples t^o and the sample mean m^o (Eq. (5)–(6) in Sec. III-A). Then, each node locally centers its own data set as $\overline{Y}_k \triangleq Y_k - m^o \mathbb{1}_{t_k}^\top$, and computes its local sample covariance matrix as $\Sigma_k \triangleq \overline{Y}_k \overline{Y}_k^\top$. Let $\mathsf{vec}(\cdot)$ be the operator that stacks the columns of a matrix one above the other, such that $\sigma^o = \mathsf{vec}(\Sigma^o)$ and $\sigma_k = \mathsf{vec}(\Sigma_k)$ denote the vectors of length D^2 with all the entries of the global and local covariance matrices, respectively; and $\Sigma^o = \mathsf{vec}^{-1}(\sigma^o)$. Introduce the network matrix of size $N \times D^2$ with rows the vectorized local covariance matrices $\Sigma = \begin{bmatrix} \sigma_1^\top \\ \cdots \\ \sigma_N^\top \end{bmatrix}$. Now, by applying consensus on Σ, the nodes can approximate the global total covariance matrix:

$$\chi_k\left(\frac{1}{t^o}\Sigma\right) \approx \frac{1}{N}\sum_{k=1}^{N}\frac{1}{t^o}\Sigma_k \triangleq \Sigma^o \qquad (8)$$

Finally, the nodes decompose their estimate of the global covariance matrix to obtain its own approximation to the global principal subspace. Depending on the number of consensus iterations employed during each consensus loop, this subspace will be arbitrarily close to that spanned by the dominant eigenvectors of Σ^o. The method proposed in [5], [6], denoted Consensus Based Distributed PCA, is summarized in Algorithm 1, where $\widehat{B}_k$ is the local estimate of the principal subspace B.

Numerical experiments: We use the "Lena" image at 512×512 resolution and 8 bit grayscale. The image has been divided into 441 blocks of 24×24 pixels. These blocks have been vectorized (aggregating columns) as points lying in a $D = 576$ dimensional subspace. We simulate a large and sparse network of 200 nodes with average degree 5 (see Figure 1). Each node randomly draws between 1 and 6 samples. After

Algorithm 1 Consensus Based Distributed PCA

1: **INPUT** t_k, Y_k
2: $\widehat{t}_k^o \leftarrow \chi_k(t)$ *consensus*
3: $m_k \leftarrow \frac{1}{t_k^o}\sum_{i=1}^{t_k} y_{k,i}$ *local*
4: $\widehat{m}_k^o \leftarrow \chi_k(m)$ *consensus*
5: $\overline{Y}_k \leftarrow Y_k - \widehat{m}_k^o \mathbb{1}_{t_k}^\top$ *local*
6: $\Sigma_k \leftarrow \overline{Y}_k \overline{Y}_k^\top$ *local*
7: $\widehat{\sigma}_k^o \leftarrow \chi_k(\frac{1}{t_k^o}\Sigma)$ *consensus*
8: $\widehat{B}_k \leftarrow P$ largest eigenvectors of $\widehat{\Sigma}_k^o = \mathsf{vec}(\widehat{\sigma}_k^o)^{-1}$ *local*
9: **OUTPUT** C_k

the iterative collaboration, each node will locally compute the $P = 20$ first principal components of its estimate of the global sample covariance matrix. Figure 3 displays the restored image after compression. The first row shows the result achieved by two random nodes in isolation. In the second row, we see a noticeable improvement achieved after 50 consensus rounds.

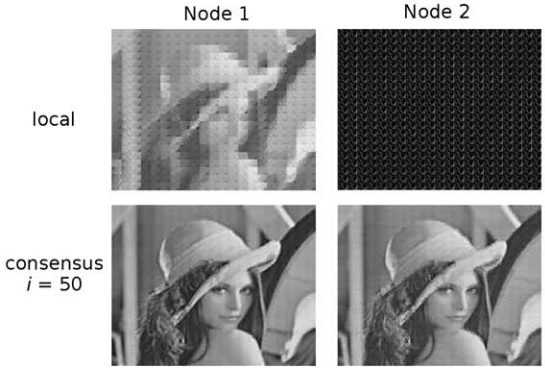

Fig. 3. Performance of PCA in isolation vs. cooperation via consensus

C. Application 2: Distributed classification with LDA

In this subsection, we apply the same idea of distributed moment estimation to a standard feature extraction method for classification, named Linear discriminant analysis (LDA). Consider a dataset in which the points belong to one of C classes. Let T_c the number of samples that belong to class c. We use the average distance between the mean of the classes as a measure of the between-class covariance:

$$\Sigma_b^o = \frac{1}{C}\sum_{c=1}^{C}(m_c^o - m^o)(m_c^o - m^o)^T \tag{9}$$

where $m_c^o = \frac{1}{T_c}\sum_{i\in c} y_i$ is the mean of the points in class c. Let Σ^o be the total sample covariance matrix in (7). LDA obtains the optimal transformation which minimizes the within-class distance while maximizing the between-class distance simultaneously. In order to find that transformation there are several criteria that can be optimized [7], such as

$$J(B) = trace\left\{(B^T\Sigma^o B)^{-1}B^T\Sigma_b^o B)\right\} \tag{10}$$

Maximizing (10) can be seen as a general eigenvalue problem (GEP), and the LDA solution is given by the P largest eigenvectors of

$$B = eigenvectors\left\{(\Sigma^o)^{-1}\Sigma_b^o)\right\} \quad (1,\ldots,P) \tag{11}$$

Under the assumption that the data classes follow Gaussian distributions with equal covariances, matrix B linearly transforms the original data space into a lower dimensional feature space such that, the linear discriminants are optimal.

In reference [4], the authors and Belanovic proposed several schemes to obtain B in a distributed fashion. The simplest one consists of all nodes approximating the global sample covariance matrix, as well as the between class and within classes scatter matrices in a similar manner as explained above for PCA. Then, they can locally compute their own estimate of B using (11) over these estimates.

Numerical experiments: Consider a Euclidian network of 100 nodes with an average degree of 5 that is provided with a distributed data set of 600 2-dimensional samples in 3 classes, with each class being Gaussian distributed with the same covariance but different mean. Each node randomly receives between 0 and 4 samples from the data set. Note that a certain number of nodes in the network will have no local samples for some classes, this is not an obstacle for the proposed algorithm. Figure 4 shows the subspace achieved in isolation

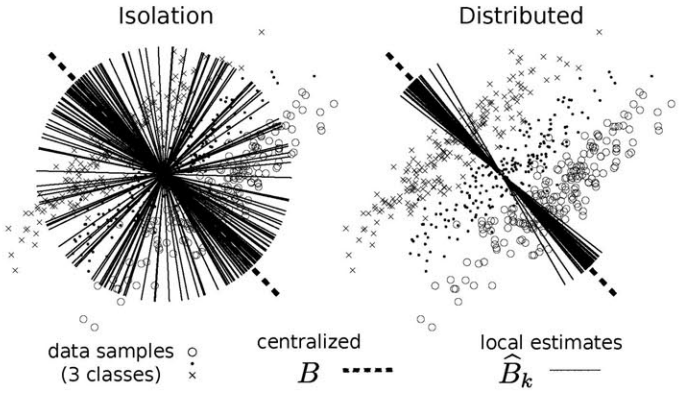

Fig. 4. Performance of LDA in isolation (left) vs. cooperation after 10 consensus rounds (right).

(left) and after 10 consensus rounds (right). The subspace B found by the centralized LDA algorithm, which has access to the entire data set, is shown as a thick dashed line. Cooperation among the nodes clearly leads to significant alignment of local estimates $\widehat{B}_k$ (thin solid lines) to the centralized solution.

IV. Distributed stochastic approximation with application to stochastic control

In this section, we review another approach to signal processing in networks based on distributed stochastic approximation. It has been shown in [3], [8] that when the nodes perform (1a)-(1b), they can find a linear combination of the extrema of the functions $f_k(w)$. In particular, if $f_k(w) \triangleq \nabla_w J_k(w)$, then the nodes find a Pareto optimal point, such

that its position in the Pareto surface is determined by the network topology (in particular by the combination matrix A). Chen, Sayed and the authors of this work used this result to solve a stochastic-control/reinforcement-learning problem in a distributed manner even when the individual agents have only access to some regions of the state-space [9].

Consider a single agent operating in an environment that can be modeled as a Markov-decision-process (MDP), which is characterized by a finite countable set of states $\mathbb{S}$ of size $S \triangleq |\mathbb{S}|$; a finite and countable set of actions $\mathbb{A}$; the kernel of transition probabilities $\mathcal{P}(s'|s,a)$ of going from one state s to another state s', given an action a; and the reward function $r : \mathbb{S} \times \mathbb{A} \times \mathbb{S} \to \Re$ that the agent wants to predict, which is associated with every transition. The agent needs to evaluate whether a policy π is good or bad in terms of the long term reward after following such policy. More formally, the agent aims to predict the expected accumulated reward when it starts in some state s and follows π afterward, this is given by the "value function" of policy π:

$$v^{\pi}(s) \triangleq \mathbb{E}_{\pi,\mathcal{P}} \left[\sum_{i=1}^{\infty} \gamma^{i-1} r(i) | s(0) = s \right] \tag{12}$$

where $\gamma \in [0,1)$ is a discount factor that weights higher the reward in the near future. Nevertheless, the agents do not know anything of the environment (e.g., no state-transition model is available), so they have to predict the optimal policy by trial and error. In addition, we assume the state-space is large, so the agent represents the states by feature vectors, $x_s \in \Re^D$ with $D \ll S$. We approximate the original value function $v^{\pi}(s)$ as a linear approximation of x_s, for some parameter vector $w \in \Re^D$: $v^{\pi}(s) \approx x_s^{\top} w$. Thus, the prediction problem becomes learning w. Let $X \triangleq \text{col}\{x_1^{\top}, \ldots, x_S^{\top}\} \in \Re^{S \times M}$ be the matrix of size $S \times D$ formed by stacking $x_s^{\top}$, on top of each other. Then, the linear approximation can be expressed in vector form as $v^{\pi} \approx Xw$ where $v^{\pi} \triangleq [v^{\pi}(1), \ldots, v^{\pi}(S)]^{\top} \in \Re^S$. As it is explained in [9], [10], w can be approximated by solving the following optimization problem:

$$\underset{w}{\text{minimize}} \quad J_{\text{PB}}(w) \triangleq \|\Pi(r^{\pi} + \gamma P^{\pi} X w) - Xw\|_M^2 \tag{13}$$

where M is a diagonal matrix with the stationary distribution of the states.

In the distributed, multiagent scenario, each agent has only access to small but complementary regions of the state space, such that every state is visited by at least one agent. The agents can benefit from each other experience just by using (1b)–(1b), i.e., sharing their local estimates $w_{k,i}$, without having to exchange any data sample (i.e., reward, state or action). Thus, although each agent will have a particular M_k, possibly with some zeros in the diagonal, it will effectively use the global $M = \sum_{k=1}^{N} M_k$ with all diagonal entries being positive. Chen, Sayed and ourselves showed in [10] that this multiagent problem can be posed as finding the saddle-point of the following Lagrangian:

$$L(\theta, w) = \theta^{\top} X^{\top} \sum_{k=1}^{N} M_k \left(\frac{1}{2} X\theta - r^{\pi} + (I - \gamma P^{\pi}) X w \right) \tag{14}$$

Then, we proposed a distributed stochastic version of the Arrow-Hurwicz algorithm, in which the agents compute distributed gradient descent and ascent in the primal and dual variables of (14), namely, θ and w, respectively. Hence, we use (1b)–(1b) twice:

$$\widehat{\theta}_{k,i+1} = \theta_{k,i} - \mu\big(x_{k,i}^{\top}\theta_{k,i} + \delta_{k,i+1}^{\top} w_{k,i} - r_k(i+1)\big) \tag{15a}$$

$$\theta_{k,i+1} = \sum_{l \in \mathcal{N}_k} a_{lk}\, \widehat{\theta}_{l,i+1} \tag{15b}$$

$$\widehat{w}_{k,i+1} = w_{k,i} + \mu \delta_{k,i+1} x_{k,i}^{\top} \theta_{k,i} \tag{15c}$$

$$w_{k,i+1} = \sum_{l \in \mathcal{N}_k} a_{lk}\, \widehat{w}_{l,i+1} \tag{15d}$$

Numerical experiments: We conclude this paper comparing the results achieved by a network of 15 agents foraging and avoiding a predator in a 2D-world (see details in [10]). Each agent samples a small region of the state-space, hence, the non-cooperative algorithm may diverge. On the other hand, when the agents cooperate they approach the same solution as a centralized architecture (see Figure 5).

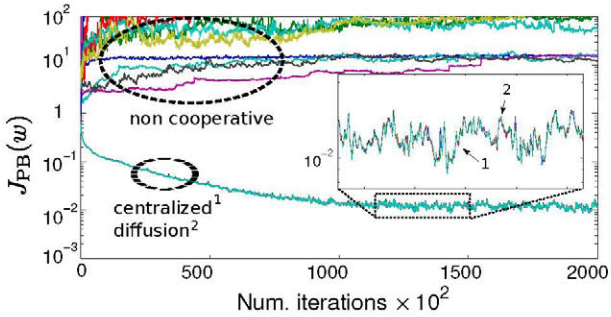

Fig. 5. Error for centralized[1], diffusion[2] and non-cooperative solutions.

REFERENCES

[1] A. H. Sayed, "Diffusion adaptation over networks," in Academic Press Library in Signal Processing, R. Chellapa and S. Theodoridis, Eds. Elsevier, 2014, vol. 3, pp. 323–454. Also available as arXiv:1205.4220v1, May 2012.

[2] F. Garin and L. Schenato, "A Survey on Distributed Estimation and Control Applications Using Linear Consensus Algorithms," in *Networked Control Systems*. Springer, 2011, vol. 406, pp. 75–107.

[3] J. Chen and A. H. Sayed, "On the Learning Behavior of Adaptive Networks - Part I: Transient Analysis," *CoRR*, vol. abs/1312.7581, 2013.

[4] S. V. Macua, P. Belanovic, and S. Zazo, "Distributed Linear Discriminant Analysis," in *IEEE ICASSP*, 2011.

[5] P. Belanovic, S. V. Macua, and S. Zazo, "Distributed static linear gaussian models using consensus," *Neural Networks*, vol. 34, pp. 96–105, 2012.

[6] S. V. Macua, P. Belanovic, and S. Zazo, "Consensus-Based Distributed Principal Component Analysis in Wireless Sensor Networks," in *IEEE SPAWC*, 2010.

[7] F. De la Torre, "A least-squares framework for component analysis," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 6, pp. 1041–1055, June 2012.

[8] J. Chen and A. H. Sayed, "Diffusion adaptation strategies for distributed optimization and learning over networks," *IEEE Transactions on Signal Processing*, vol. 60, no. 8, pp. 4289–4305, Aug. 2012.

[9] S. V. Macua, J. Chen, S. Zazo, and A. H. Sayed, "Cooperative off-policy prediction of Markov decision processes in adaptive networks," in *IEEE ICASSP*, 2013, pp. 4539–4543.

[10] ——, "Distributed policy evaluation under multiple behavior strategies," *CoRR*, vol. abs/1312.7606, 2013.